Math. Control Signals Systems (1989) 2: 95–107

Mathematics of Control, Signals, and Systems
© 1989 Springer-Verlag New York Inc.

On the Control of Discrete-Event Dynamical Systems*

John N. Tsitsiklis†

Abstract. We study a class of problems related to the supervisory control of a discrete-event system (DES), as formulated by Ramadge and Wonham, and we focus on the computational effort required for their solution. While the problem of supervisory control of a perfectly observed DES may be easily solved by dynamic programming, the problem becomes intractable (in the sense of complexity theory) when imperfectly observed systems are considered.

Key words. Discrete-event systems, Supervisory control, Computational complexity, Partial observations.

1. Introduction

Discrete-event systems (DES) have been introduced by Ramadge and Wonham [RW2]. Roughly speaking a DES is a discrete-time dynamical system such that, for each state, a number of different transitions may occur. Furthermore, it is assumed that there is a possibility for control action through a supervisor which, at any given point in time, may prohibit certain transitions from occurring. It is then natural to consider the problem of designing such a supervisor satisfying certain specifications. Loosely speaking, the specifications that have been considered in the literature amount to a requirement that the supervisor prohibits from occurring certain (undesirable) sequences of events, while at the same time allowing some other (desirable) sequences of events to occur. Naturally, the supervisor design problem changes according to the different assumptions made concerning the information available to the supervisor; for example, the supervisor may have full knowledge of the state of the DES (perfect information), or it may have access only to some partial information on the state. Decentralized supervision by a set of noncommunicating supervisors, each one possessing partial state information, leads to another class of design problems.

A DES is very similar to a discrete-time Markov chain, except that there are no assumptions about the probabilities of the state transitions. For this reason, the supervisor design problem is a little different from the traditional problems of

* Date received: April 15, 1987. Date revised: December 14, 1987. Research supported by the Army Research Office (Grant No. DAAL03-86-K-0171) and by an NSF PYI award, with matching funds from Bellcore Inc.

† Room 35-214, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, U.S.A.

Markov decision theory, for which dynamic programming provides a solution [B]. On the other hand, Markov decision problems and the supervisor design problem for a DES are not completely unrelated. Consider the supervisor design problem under a constraint that certain states must be avoided. We may assign an infinite cost to the states to be avoided, zero cost to the remaining states, and assign arbitrarily a positive probability to each possible transition out of given state, thus defining a Markov decision problem. These two problems are closely related because any supervisor for the DES satisfying the specifications corresponds to a finite cost policy for the Markov decision problem. Other types of specifications for the supervisor of the DES may be easily incorporated into the cost function of a corresponding Markov decision problem; see Section 3 for a more detailed exposition.

From these remarks, it is natural to suspect that the types of problems which can be solved realistically within the DES framework (from a computational point of view) correspond to easily solvable problems in Markov decision theory. Thus, in the light of available results [PT], it should be expected that problems with partial information are algorithmically intractable. One of the aims of this paper is to justify and give a precise content to the above statement.

The paper is organized as follows. In Section 2 we introduce the definitions, notation, and terminology to be employed. In Section 3 we provide a brief background for the case of perfect information. In Section 4 we consider a variety of supervisor design problems when only partial information is available. While a special case of this problem, studied in [CDFV], is shown to be tractable, a number of negative results are derived for several interesting problems.

2. Preliminaries

A DES G can be defined [RW2] as a quadruple $G = (Q, \Sigma, \delta, q_0)$, where Q is a finite set (state space), q_0 is an element of Q (initial state), Σ is a finite alphabet (used to label possible transitions between states, also called events), and δ is a partial function (i.e., which is defined only on a subset of its domain) from $Q \times \Sigma$ into Q, which describes the dynamics of the system. The interpretation of δ is the following: if $\delta(q, \sigma)$ is defined for some given $q \in Q$, $\sigma \in \Sigma$, then it is possible that, starting from q, a transition carrying the label σ takes place and, in that case, the next state is equal to $\delta(q, \sigma)$.

Actually the definition usually given is somewhat more involved because it includes a special set Q_m of marked states ("accepting states," in the language of automata theory). We chose to omit them from the definition in order to simplify notation. Let us just mention here that the computational complexity of the problems considered in this paper is unaffected by the exclusion of marked states from the DES model.

Occasionally we will find the following notation a little more convenient: for any $q \in Q$, we are given a set $\Sigma(q) \subset \Sigma$, the set of possible transition labels out of state q; in that case, δ is a function (total, rather than partial) defined on the set $\bigcup_{q \in Q}(\{q\} \times \Sigma(q))$. In traditional systems-theoretic terminology, we are dealing with a dynamical system with state q, subject to uncertain disturbances σ; the system obeys the dynamical equation

$$q(t+1) = \delta(q(t), \sigma(t)); \qquad q(0) = q_0, \tag{1}$$

and the disturbances $\sigma(t)$ are constrained to satisfy

$$\sigma(t) \in \Sigma(q(t)) \quad \text{for all } t. \tag{2}$$

Here t is a discrete variable used to index events and need not be related to "real time."

A *string* is the concatenation of a finite (possibly empty) sequence $(\sigma(0), \dots, \sigma(t))$ of elements of Σ. Let ε denote the empty string and let Σ^* denote the set of all strings. We extend the function δ to a partial function from $Q \times \Sigma^*$ into Σ by means of the following recursive definition: $\delta(q, \varepsilon) = q$ and $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$, if $\delta(q, s)$ is defined, and $\sigma \in \Sigma(\delta(q, s))$. In particular, $\delta(q, s)$ is equal to the current state if the initial state is equal to q and the sequence of transitions represented by the string s has occurred, assuming that this sequence of transitions is allowed by (2).

Any subset of Σ^* is called a *language*. We define $L(G)$, the language generated by G, as the set of all strings s such that $\delta(q_0, s)$ is defined. Notice that $L(G)$ always contains the empty string.

We now provide for the possibility of controlling a DES. We assume that the set Σ is partitioned into two disjoint subsets Σ_u and Σ_c. The set Σ_c is interpreted as the set of events which can be disabled. We define a *supervisor* for G as a function $\gamma\colon \Sigma^* \mapsto 2^\Sigma$, such that $\gamma(s) \supset \Sigma_u$, for all $s \in \Sigma^*$. The set $\gamma(s)$ is the set of events that are allowed by the supervisor to occur (not disabled), as a function of the string s of past events. Accordingly, in the presence of a supervisor γ, we obtain a new dynamical system whose state again satisfies (1), but the constraint (2) now becomes

$$\sigma(t) \in \gamma(\sigma(0)\cdots\sigma(t-1)) \cap \Sigma(\delta(q_0, \sigma(0)\cdots\sigma(t-1))) = \gamma(\sigma(0)\cdots\sigma(t-1)) \cap \Sigma(q(t)). \tag{3}$$

A DES G together with a supervisor γ, are called a *supervised system*. Given a supervised system (G, γ), we define the language $L(G, \gamma)$ as the set of all strings in Σ^* that can be generated by that system. More formally, $L(G, \gamma)$ is the set of all strings $\sigma(0)\cdots\sigma(T) \in L(G)$, which also satisfy (3), for each $t \leq T$, the empty string being included. Let us point out that we could require γ to be a partial function defined only on the subset of Σ^* consisting of those strings whose occurrence is possible, that is on $L(G, \gamma)$. However, we assume that γ is total, to simplify notation and the discussion.

In general, a supervisor need not have access to the entire string of past events; this may place a restriction on the set of supervisors under consideration. We then say that *partial information*, as opposed to *perfect information*, prevails. There are several possible ways of modeling partial information and we follow here the formulation in [CDFV]. Consider a function $M\colon \Sigma \mapsto \Pi \cup \{\varepsilon\}$, where Π is another finite alphabet and where ε denotes the empty string. Following [CDFV], call such a function a *mask*. We interpret $M(\sigma(t))$ as the information provided to the supervisor on the value of $\sigma(t)$. However, the possibility that $M(\sigma(t))$ equals the empty string allows a situation where the supervisor does not learn that a transition has occurred. (In particular, a supervisor does not know, in general, the value of the time variable

t. We extend M to a mapping from Σ^* into Π^* by letting $M(\sigma(0)\cdots\sigma(t))$ be the concatenation of $M(\sigma(0)), \ldots, M(\sigma(t))$. A supervisor γ is called an *M-supervisor* if there exists some function $\gamma_M\colon \Pi^* \mapsto 2^\Sigma$ such that $\gamma(s) = \gamma_M(M(s))$ for all $s \in \Sigma^*$.

A special class of supervisors is the class of *state feedback supervisors*. A supervisor γ belongs to this class if there exists a function $\gamma_F\colon Q \mapsto 2^\Sigma$ such that $\gamma(s) = \gamma_F(\delta(q_0, s))$ for all $s \in L(G)$. Such supervisors are easily implemented in the case of perfect information because the current state can be reconstructed from the knowledge of all past events but this is, in general, impossible in the case of partial information.

Another interesting class of supervisors is the set of *finite state* supervisors. A supervisor γ belongs to this class if there exists a DES $\hat{G} = (\hat{Q}, \hat{q}_0, \Sigma, \hat{\delta})$ and a function $\gamma_R\colon \hat{Q} \mapsto 2^\Sigma$ such that: (a) $\hat{Q}$ is a finite set; (b) $\hat{\delta}$ is a total function; (c) $\gamma(s) = \gamma_R(\hat{\delta}(\hat{q}_0, s))$. Any such $\hat{G}$, together with the mapping γ_R, is called a *finite state realization* of γ.

We note that if Q is finite then any state feedback supervisor is also a finite state supervisor. The corresponding DES $\hat{G}$ is just a duplicate of the supervised DES G; it keeps track of the state $q(t)$ of G and at each time instance it chooses its supervisory action appropriately.

Let us briefly recall concepts from complexity theory to be used later. We will only consider *decision problems*, that is problems in which a yes/no question is posed. As usual, P (resp. NP, PSPACE) stands for the class of decision problems solvable by a polynomial-time (resp. nondeterministic polynomial, polynomial memory) algorithm. A problem is NP-complete (resp. PSPACE-complete) if it belongs to NP (resp., PSPACE) and any problem in NP (resp. PSPACE) may be reduced to it via a polynomial-time transformation. A problem is NP-hard, (resp. PSPACE-hard) if some NP-complete (resp. PSPACE-complete) may be reduced to it by a polynomial-time transformation. We have $P \subset NP \subset PSPACE$ and it is widely conjectured that both inclusions are proper. If this conjecture is true, then there do not exist any polynomial-time algorithms for NP-complete, NP-hard, or PSPACE-complete problems. The reader is refered to [PS] for a more detailed and precise exposition of these concepts.

3. Supervisor Design: Perfect Information

A representative supervisor design problem introduced in [RW2] is the following: given three DESs G, G_1, G_2, employing the same alphabet Σ, we are asked to determine whether there exists a supervisor γ such that

$$L(G_1) \subset L(G, \gamma) \subset L(G_2).$$

This problem has been solved in [RW2] and [WR] and polynomial upper bounds on the required computations are given in [RW1]. We outline below a different way of getting to these results, by framing the problem in a Markov decision context.

Any DES may be modified so that the corresponding transition function is total. In particular, given a DES $G = (Q, q_0, \Sigma, \delta)$, we define a new DES $G' = (Q \cup \{*\}, q_0, \Sigma, \delta')$, where $*$ is a new (trap) state. We let $\delta'(q, \sigma) = \delta(q, \sigma)$, whenever $\delta(q, \sigma)$ is defined, and $\delta'(q, \sigma) = *$, otherwise, Notice that $\delta'(q, s) = *$ if and only if $s \notin L(G)$.

We assume that all three DESs introduced above (G, G_1, G_2) have been so augmented.

Consider a new DES consisting of the augmented versions of G, G_1, G_2, running simultaneously, under the influence of the same input sequence $(\sigma(0), \sigma(1), \ldots)$ and starting from their respective initial states. Let $q'(t)$, $q'_1(t)$, $q'_2(t)$ denote their respective states at time t. We now interpret our supervisor specifications as state constraints. The inclusion $L(G_1) \subset L(G, \gamma)$ requires that if $(\sigma(0) \cdots \sigma(t-1)) \in L(G_1)$ (that is, if $q'_1(t) \neq *$), then $(\sigma(0) \cdots \sigma(t-1)) \in L(G, \gamma)$ (that is, $q'(t) \neq *$). Therefore, this constraint is captured by assigning infinite cost to any state (q', q'_1, q'_2) of the composite DES such that $q'_1 \neq *$ and $q' = *$. Similarly, the constraint $L(G, \gamma) \subset L(G_2)$ is equivalent to assigning infinite cost to any state (q', q'_1, q'_2) such that $q' \neq *$ and $q'_2 = *$. Clearly, the original supervisor design problem has a solution if and only if there exists a control law for the above-defined composite system under which the cost (starting from the appropriate initial state) is finite, for any possible sequence of events. This would be a standard dynamic-programming problem: the only difference is that we are dealing with a worst-case (minimax) criterion instead of an expected cost criterion. However, it is well known that the dynamic-programming algorithm is equally applicable to such minimax problems and has polynomial computational requirements [B]. As this is a well-known algorithm, we omit a detailed description. In fact, the structure of this problem is so simple that the dynamic-programming algorithm simplifies to a connectivity test; still, it is important to realize that the computational requirements of this problem are polynomial because it is a special case of a control problem solvable by dynamic programming.

An alternative design criterion that has been proposed is as follows: the objective now is to find a supervisor γ such that $L(G, \gamma)$ is maximal, subject to the constraint $L(G, \gamma) \subset L(G_2)$. (Existence and uniqueness of a maximal $L(G, \gamma)$ has been proved in [RW2].) We point out that this problem can also be formulated as a dynamic-programming problem. We do not provide the details, but the key idea is the following: we express the requirement $L(G, \gamma) \subset L(G_2)$ as a state constraint (as with the previous problem) and we enforce maximality of the supervisor by introducing a penalty term which increases with the number of disabled transitions at each stage.

Notice that we are not suggesting that a reformulation as a traditional control problem be used in order to obtain an algorithm for supervisor design. The value of the above arguments is that they prove with minimal effort that these problems are polynomially solvable.

We finally mention the problem of optimal supervisor reduction. In this problem we are given a DES G, and by some design procedure we have chosen a supervisor γ. Suppose, furthermore, that γ is a finite-state supervisor. As there exist infinitely many alternative finite-state realizations of such a supervisor, we are interested in a realization with a minimal number of states. This problem has been studied in [VW]. This reference provides an algorithm for constructing such a minimal supervisor. However, this algorithm requires, in general, a computational effort which is exponential in the number of states of G. While a polynomial algorithm would be desirable, this is very unlikely, because the problem under consideration is NP-complete. A proof can be found in [T]. The same result can also be obtained by realizing that the supervisor reduction problem is closely related to the problem of state reduction in incompletely specified finite-state machines which is known to be NP-complete [P]. (We thank an anonymous referee for this observation.)

4. Supervisor Design: Partial Information

Let M be a mask, as defined in Section 2. We consider here certain supervisor design problems, similar to those considered in Section 3 except for the additional requirement that the supervisor designed is an M-supervisor. The simplest such problem addresses the question whether there exists some M-supervisor γ such that $L(G, \gamma) = L(G_1)$, where G_1 is a given DES. This problem has been studied in [CDFV] where the following result is proved:

Proposition 4.1. *Given two DESs G, G_1, such that $L(G_1) \subset L(G)$, and a mask M: $\Sigma \mapsto \Pi \cup \{*\}$, there exists an M-supervisor γ such that $L(G, \gamma) = L(G_1)$ if and only if the following two properties hold:*

(a) *There exists a supervisor γ such that $L(G, \gamma) = L(G_1)$.*
(b) *If $s, s' \in L(G_1)$, $\sigma \in \Sigma_c$, $s\sigma \in L(G_1)$, $s'\sigma \in L(G)$, and $M(s) = M(s')$, then $s'\sigma \in L(G_1)$.*

We now show that the conditions in Proposition 4.1 may be tested in a computationally efficient way:

Proposition 4.2. *There exists a polynomial-time algorithm (polynomial in the cardinalities of the state spaces of G and G_1) for deciding whether the conditions in Proposition 4.1 are valid.*

Proof. (a) Testing this condition is equivalent to solving the first problem of Section 3, for the special case where $G_1 = G_2$, and can therefore be done in polynomial time.

(b) Let $G = (Q, q_0, \Sigma, \delta)$, $G_1 = (Q_1, q_0^1, \Sigma, \delta_1)$, and let $\Sigma(q)$, $\Sigma_1(q^1)$ be the allowed transitions under G, G_1, respectively, when the current state is q, q^1, respectively. Consider the following game: we start at the "state" (q_0^1, q_0^1, q_0). In general, at each point in time our state is a triple $(q^1, q^2, q^3) \in Q_1 \times Q_1 \times Q$ and we have the following options: choose some $\sigma \in \Sigma_1(q^1)$, choose some $\sigma' \in \Sigma(q^3)$, or choose both a σ and a σ' as above. The rules of the game are as follows: if we have chosen some $\sigma \in \Sigma_1(q^1)$ such that $M(\sigma) \neq \varepsilon$, then we must simultaneously choose some $\sigma' \in \Sigma(q^3)$ such that $M(\sigma) = M(\sigma')$. Similarly, if we have chosen some $\sigma' \in \Sigma(q^3)$ such that $M(\sigma') \neq \varepsilon$, then we must simultaneously choose some $\sigma \in \Sigma(q^1)$ such that $M(\sigma) = M(\sigma')$. After we make our choices, the states move as follows: q^1 does not change if no σ is chosen; otherwise it moves to $\delta_1(q^1, \sigma)$. The states q^2, q^3, do not change if no σ' is chosen; otherwise, q^3 moves to $\delta(q^3, \sigma')$ and q^2 moves to $\delta_1(q^2, \sigma')$, except if $\sigma' \notin \Sigma_1(q^2)$, in which case the game terminates. We win if and only if the game terminates and the σ' causing the termination is equal to the last σ chosen and belongs to Σ_c.

Suppose that there exists a winning strategy in this game. Let $\bar{s} = (\sigma(0) \cdots \sigma(m))$ and $\bar{s}' = (\sigma'(0) \cdots \sigma'(n))$ be the strings of σ's and σ''s used in our winning strategy. Let $s = (\sigma(0) \cdots \sigma(m-1))$ and $s' = (\sigma'(0) \cdots \sigma'(n-1))$. Since we win, we have $\sigma(m) = \sigma'(n) = \sigma$ for some $\sigma \in \Sigma_c$. Therefore, $\bar{s} = s\sigma$ and $\bar{s}' = s'\sigma$. Because of the rules of the game, we have $M(s) = M(s')$. Furthermore, $s\sigma \in L(G_1)$, because at each time we choose a σ belonging to $\Sigma_1(q^1)$. Similarly, since the game was not terminated before $\sigma'(n)$ was chosen, it follows that at each choice except for the last one we had $\sigma' \in \Sigma_1(q^2)$ and therefore $s' \in L(G_1)$. Since $L(G_1) \subset L(G)$, we also have $s' \in L(G)$ and since $\sigma'(n) \in \Sigma(q^3)$, it follows that $s'\sigma \in L(G)$. On the other hand, since we have won the game, we must have $\sigma \notin \Sigma_1(q^2)$. Therefore, $s'\sigma \notin L(G_1)$ and condition (b) is violated.

The above argument can be reversed: if there exist s, s', σ for which condition (b) is violated, then we use them to define a winning strategy in the above game. Thus (b) holds if and only if there exists no winning strategy in our game.

It follows that it is sufficient to devise an algorithm which determines if there exists a winning strategy for the above game. This is just a deterministic optimal control problem on a finite-state space, with state-dependent control constraints. Dynamic programming applies and provides a polynomial-time algorithm for solving it, which concludes the proof. ∎

Proposition 4.2 is a positive result, especially given the fact that control problems with partial information are often intractable. Notice, however, that we have only found a way for deciding whether an M-supervisor exists, but we do not yet have an efficient method for constructing it. It is shown in [CDFV] that if there exists an M-supervisor γ such that $L(G, \gamma) = L(G_1)$ and if G, G_1 have finite-state space, then the supervisor γ may be chosen to be a finite-state supervisor. A reasonable choice for the state space of γ is to let it be equal to the power set of $Q \times Q_1$, where Q, Q_1 are the state spaces of G, G_1, respectively. With this choice, a state of the supervisor indicates the set of all states of G, G_1, which are possible, given the available information. However, such a state space has cardinality which is exponential in the size of the state space of G and therefore an exponential amount of computational resources is required to construct it. Given the positive result in Proposition 4.2, we might hope that a supervisor with a polynomial-state space may always be found. The family of examples provided below shows that this is not so.

Example. Let us fix some positive integer n. The DES G to be supervised has an associated alphabet $\Sigma = \{u_1, \ldots, u_n\} \cup \{d_1, \ldots, d_n\} \cup \{0, 1\} \cup \{\alpha_1, \ldots, \alpha_n\}$. The language $L(G)$ generated by G consists of all prefixes of strings of the form $(\sigma(0) \cdots \sigma(n+2))$ with the following properties: $\sigma(0) \in \{u_1, \ldots, u_n\} \cup \{d_1, \ldots, d_n\}$; $\sigma(i) \in \{0, 1\}$ for $i = 1, \ldots, n$ and $i = n+2$; $\sigma(n+1) \in \{\alpha_1, \ldots, \alpha_n\}$. Furthermore, if $\sigma(0) = u_k$, then $\sigma(k) = 1$ and $\sigma(n+1) = \alpha_k$; also, if $\sigma(0) = d_k$, then $\sigma(k) = 0$ and $\sigma(n+1) = \alpha_k$.

Notice that $L(G)$ is a finite language and may therefore be generated by a finite-state DES. In fact, we may choose the state space of G to be as small as $O(n^2)$. This is done as follows: except for an initial state q_0, we let the other states be pairs (x, t) where t counts the number of transitions made so far and where x is equal to $\sigma(0)$. Figure 1 presents a state transition diagram for the case $n = 3$.

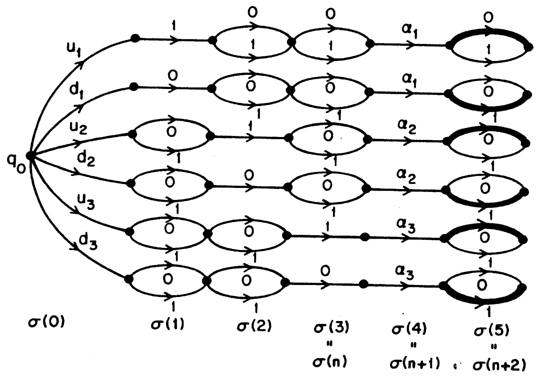

Fig. 1

We observe G through a mask M defined as follows: $M(u_k) = M(d_k) = \varepsilon$, for all k, and $M(\sigma) = \sigma$ if $\sigma = \alpha_k$ or $\sigma \in \{0, 1\}$. Let our target language $L(G_1)$ be the same as $L(G)$ except that if $\sigma(0) = u_k$, we then require that $\sigma(n + 2) = 1$, and if $\sigma(0) = d_k$, we require $\sigma(n + 2) = 0$. Notice that G_1 is a finite-state DES: it coincides with G except that we delete one of the two possible transitions out of any state that can be reached after exactly $n + 2$ transitions. (The deleted transitions correspond to the thicker lines in Fig. 1.)

It is easy to see that there exists an M-supervisor such that $L(G, \gamma) = L(G_1)$: the supervisor remembers $\sigma(1), \ldots, \sigma(n)$. When $\sigma(n + 1)$ occurs, the supervisor observes α_k, for some k, and retrieves the value of $\sigma(k)$. If $\sigma(k) = 1$ (resp. 0) it decides that the unobserved transition $\sigma(0)$ was equal to u_k (resp. d_k), and decides accordingly which transition to suppress. This supervisor uses $O(2^n)$ states, since at time $n + 1$ it remembers n bits of information and intuition suggests that no reduction of its state space is possible. We prove this formally. For any string $s = (\sigma(1) \cdots \sigma(n))$, let $g(s)$ denote the state of the supervisor before $\sigma(n + 1)$ is observed. The transition which is not disabled after $\sigma(n + 1)$ is observed is therefore a function of $\sigma(n + 1)$ and $g(s)$. Therefore, there exists some function f such that:

(i) $f(g(s), \alpha_k) = 1$ if $\sigma(k) = 1$.
(ii) $f(g(s), \alpha_k) = 0$ if $\sigma(k) = 0$.

Let s, s' be such that $s \neq s'$. Assume that s and s' differ in their kth symbol. Then we must have $f(g(s), \alpha_k) \neq f(g(s'), \alpha_k)$, which implies that $g(s) \neq g(s')$. This shows that g is a one-to-one mapping and therefore its range has cardinality 2^n. Hence, the state space of the supervisor must have cardinality at least 2^n. We have thus constructed a family of partially observed supervision problems (parametrized by n) for which an M-supervisor exists (for each n), the state space of the DES being supervised has cardinality polynomial in n, but the state space of any M-supervisor must have cardinality exponential in n. Furthermore, this happens even though there exists a supervisor (which is not an M-supervisor) with small (polynomial in n) state space.

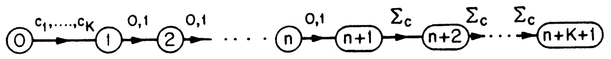

Fig. 2

The supervisor design problem of Proposition 4.1 seems to be about the only partial information problem for which something can be done in polynomial time. We justify this claim by studying three variants of the imperfect information problem, all of which are found to be algorithmically intractable.

Problem A. This problem is very similar to the one considered in Section 3. Given three finite-state DESs G, G_1, G_2, and a mask M, does there exist an M-supervisor γ such that $L(G_1) \subset L(G, \gamma) \subset L(G_2)$?

Proposition 4.3. *Unless* P = NP, *there is no polynomial-time algorithm for Problem A. (In particular, the complement of Problem A is* NP-*hard.)*

Proof. We reduce the complement of the 3SAT ("three satisfiability") problem of propositional calculus to Problem A. In the 3SAT problem we are given n literals (Boolean variables) $v_1, \ldots, v_n$, and K clauses $C_1, \ldots, C_K$, and we are asked whether there exists an assignment of truth values to the literals so that all clauses are true. We now assume that an instance of 3SAT is given and we construct an equivalent instance of Problem A.

We first describe the DES G. The set of events Σ is given by $\Sigma = \{c_1, \ldots, c_K, 0, 1, \sigma_1, \ldots, \sigma_K\}$ and the structure of G is shown in Fig. 2. In particular, $L(G)$ is the set of prefixes of all strings of the form $c_k s s'$ where $s \in \{0, 1\}^n$ and $s' \in \{\sigma_1, \ldots, \sigma_K\}^K$. We let the set Σ_c of controllable events be given by $\Sigma_c = \{\sigma_1, \ldots, \sigma_K\}$. We let $\Pi = \{0, 1, \varepsilon\}$ and we consider the mask $M: \Sigma \to \Pi$ given by $M(0) = 0, M(1) = 1, M(c_k) = M(\sigma_k) = \varepsilon$ for each k. Effectively, the supervisor observes a sequence $s \in \{0, 1\}^n$ and decides what elements of Σ_c are to be disabled afterwards. For this reason, we identify a supervisor with a function $\gamma: \{0, 1\}^n \to \Sigma_c$ where $\gamma(s)$ is the subset of Σ_c which is enabled, given that s has been observed.

We continue with the supervisor specifications. We let G_2 be the DES shown in Fig. 3. In particular, $L(G_2) \cap L(G) = L(G) - \{c_k s \sigma_1 \cdots \sigma_K : s \in \{0, 1\}^n, k = 1, \ldots, K\}$. The specification $L(G, \gamma) \subset L(G_2)$ translates to the requirement that $\gamma(s) \neq \Sigma_c$ for every $s \in \{0, 1\}^n$. To see this, suppose that $\gamma(s) = \Sigma_c$ for some s. Then the string

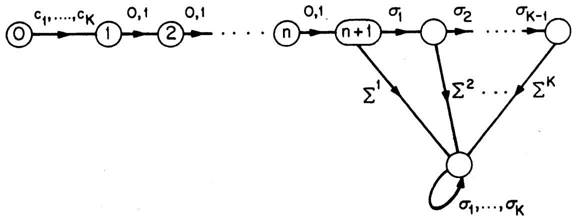

Fig. 3. The DES G_2. Here Σ^k stands for the set $\{\sigma_i | 1 \leq i \leq K \text{ and } i \neq k\}$.

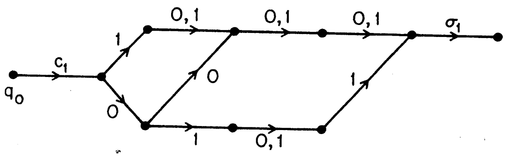

Fig. 4. The DES G_1. In this example we have $n = 4$ and a single clause ($K = 1$). The clause is $(v_1 \vee \bar{v}_2 \vee v_4)$. Notice that the state moves to the upper row of states as soon as the clause is satisfied.

$c_1 s\sigma_1 \cdots \sigma_K$ is allowed to occur, violating the requirement $L(G, \gamma) \subset L(G_2)$. Conversely, if $\gamma(s) \neq \Sigma_c$, for each s, no string of the form $c_k s\sigma_1 \cdots \sigma_K$ may occur, because at least one of the σ_k's in the substring $\sigma_1 \cdots \sigma_K$ is disabled.

Let S_k be the set of all strings $s = v_1 \cdots v_n \in \{0, 1\}^n$ such that $v_1, \ldots, v_n$ satisfy the kth clause C_k. Let $S = \bigcap_{k=1}^{K} S_k$ and notice that $S \neq \varnothing$ if and only if the given instance of 3SAT is a "YES" instance. We let the DES G_1 be such that $L(G_1)$ is equal to the set of all prefixes of strings belonging to the set

$$\bigcup_{k=1}^{K} \{c_k s\sigma_k : s \in S_k\}.$$

Such a DES G_1 is easily contructed (in polynomial time) as shown in Fig. 4. The specification $L(G_1) \subset L(G, \gamma)$ is easily seen to be equivalent to the requirement that $\sigma_k \in \gamma(s)$ for every s belonging to S_k.

We have completed the construction of the instance of Problem A. Given our earlier remarks, this is a "YES" instance if and only if there exists some $\gamma: \{0, 1\}^n \to \Sigma_c$ such that: (a) $\gamma(s) \neq \Sigma_c$ for all $s \in \{0, 1\}^n$ and (b) $\sigma_k \in \gamma(s)$ for all $s \in S_k$. We show that this is the case if and only if we started with a "NO" instance of 3SAT. Indeed suppose that we have a "YES" instance of Problem A and that γ is a supervisor satisfying (a) and (b) above. Then, for every $s \in \{0, 1\}^n$, there exists some k such that $\sigma_k \notin \gamma(s)$; using (b), for every $s \in \{0, 1\}^n$ there exists some k such that $s \notin S_k$. It follows that $S = \bigcap_{k=1}^{K} S_k$ is empty and we have a "NO" instance of 3SAT. Conversely, suppose that we have a "NO" instance of 3SAT. Then, for every $s \in \{0, 1\}^n$, there exists some k such that $s \notin S_k$ and a desired supervisor is obtained by letting $\gamma(s) = \Sigma_c - \{\sigma_k\}$. This completes the reduction. ■

It is quite likely that a stronger result can be proved, although we have not been able to do so: we conjecture that Problem A is actually PSPACE-hard. We now continue with a variation of Problem A.

Problem B. Given two finite-state DESs G and G_2, and a mask M, does there exist an M-supervisor γ such that $L(G, \gamma) \subset L(G_2)$ and such that "deadlock is impossible," meaning that we never come to a situation where all transitions out of the current state are disabled?

The "no deadlock" specification is equivalent to requiring that for every $s \in L(G, \gamma)$ there exists some $\sigma \in \Sigma$ such that $s\sigma \in L(G, \gamma)$. Such a specification cannot be expressed, in general, in the form $L(G_1) \subset L(G, \gamma)$. We show that Problem B is intractable even for the special case where $G_2 = G$. In this case, the inclusion $L(G, \gamma) \subset L(G_2)$ is trivially true and we are dealing with the following problem.

Problem B′. Given a DES G does there exist an M-supervisor γ such that deadlock is impossible?

Proposition 4.4. *Problem B′ (and, a fortiori, Problem B) is PSPACE-hard.*

Proof. The proof is patterned after the proof that the partial information Markov decision problem is PSPACE-complete [PT], with a few differences. It consists of reducing the QSAT problem (quantified satisfiability) of propositional calculus to Problem B. An instance of QSAT consists of $2n$ Boolean variables $v_1, \dots, v_{2n}$, and K disjunctive clauses $C_1, \dots, C_K$, with three literals per clause, and the problem consists of deciding whether $\exists v_1 \forall v_2 \cdots \exists v_{2n-1} \forall v_{2n}[C_1 \wedge C_2 \wedge \cdots \wedge C_K]$, where $\wedge$ denotes conjunction. (The existential quantifiers $\exists$ are alternating with the universal quantifiers $\forall$ in the above formula.) We think of this problem as a game: an opponent assigns values to the even variables $v_2, v_4, \dots, v_{2n}$ and we are to assign values to the odd variables $v_1, v_3, \dots, v_{2n-1}$, as functions of the past choices by the opponent; our objective is to have all the clauses satisfied.

Given an instance of QSAT we now construct an instance of Problem B'. The alphabet Σ is $\{u_1, \dots, u_K\} \cup \{\tau_0, \tau_1, \tau_0', \tau_1', w\}$. We let $\Sigma_c = \{\tau_0, \tau_1\}$. The mask M is such that $M(u_k) = \varepsilon$, for all k, and each τ_i, τ_i' is perfectly observed. We introduce an auxiliary function f defined by $f(\tau_0) = f(\tau_0') = 0$ and $f(\tau_1) = f(\tau_1') = 1$. Accordingly, a sequence of $2n$ transitions not involving the u_k's is associated, using the function f, with an assignment of the variables $v_1, \dots, v_{2n}$.

There is an initial state q_0 out of which any one of the transitions $u_1, \dots, u_K$ may occur, leading to states $x_1, \dots, x_K$, respectively. However, $M(u_k) = \varepsilon$, for all k, so that the state $q(1)$ after the first transition is unknown. (We may think of this as having the opponent choose a clause without revealing the choice.) The state transition diagram starting from any x_k, $k = 1, \dots, K$, does not involve any u_j's and is such that the state $q(2n + 1)$ is a "bad" state b_k (respectively, a "good" state g_k) if the assignments $v_k = f(\sigma(k))$ make the kth clause true (resp. false). (This is similar to the construction in Proposition 4.3 and is illustrated in Fig. 5 for the case $n = 2$.)

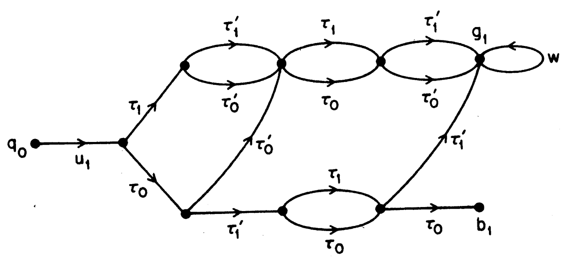

Fig. 5. An example of the reduction in Proposition 5.5. Here $n = 2$ and there is a single clause $(v_1 \vee \bar{v}_2 \vee v_4)$.

There are no transitions possible out of the bad states b_k, thus guaranteeing deadlock, whereas there is a transition w from every g_k to itself. Thus, deadlock avoidance is equivalent to guaranteeing that the state eventually reaches one of the states g_k, or, equivalently, that the clause selected by the opponent is satisfied. Since the clause selected is not known (because $M(u_k) = \varepsilon$ for all k), we should satisfy all clauses. Notice that at odd times t the supervisor may disable τ_0 or τ_1, which corresponds to assigning a truth value to the variable v_t, whereas at even times t the supervisor has no control, which corresponds to letting the opponent fix the value of v_t.

The proof is completed by showing that such a supervisor exists if and only if we are dealing with a "YES" instance of QSAT. Suppose that we have a "YES" instance of QSAT. Thus, there exists a strategy through which at any odd time $t = 1, 3, \ldots, 2n - 1$, we assign a value to v_t (equivalently, the supervisor enables exactly one of τ_0 or τ_1), as a function of past assignments, so that every clause is satisfied. This guarantees that the bad states b_k are avoided for each k.

For the converse, suppose that we have a "YES" instance of Problem B'; that is, there exists a supervisor conforming to the specifications. Since dealock is prohibited, the supervisor enables at least one of τ_0, τ_1, at any odd time. Furthermore, if a supervisor exists, then there exists a supervisor which never enables both τ_0, τ_1. (Enabling both simultaneously amounts to leaving one more variable to the control of the "opponent" and this cannot be beneficial.) The strategy of this supervisor for deciding which τ_i to enable at each odd time t then determines a strategy for assigning a value to the variable v_t so as to satisfy all clauses, which proves that we have a "YES" instance of QSAT and completes the proof. ∎

Problem C. Given a finite-state DES G, two masks $M_i\colon \Sigma \mapsto \Pi \cup \{\varepsilon\}$, $i = 1, 2$, and a finite-state DES G_1 whose symbol alphabet is Π, does there exist an M_1-supervisor γ such that $M_2(L(G, \gamma)) = L(G_1)$?

Here $M_2(L(G, \gamma))$ stands for the language consisting of the images of all strings in $L(G, \gamma)$, under the mapping M_2. This is a reasonable design criterion if our performance specifications depend on the string of transitions s through some function M_2. That is, instead of constraining s directly, we only constrain $M_2(s)$. Such a specification may be used, for example, if we want to impose a condition that the state of G eventually reaches a special state q^*, but we do not care about how it gets there.

Proposition 4.5. *Problem C is PSPACE-hard.*

Proof. We use exactly the same reduction as in the proof of Proposition 4.4. We also define M_2 by $M_2(w) = w$ and $M_2(\sigma) = \varepsilon$ for all $\sigma \neq w$. We impose the specification $M_2(L(G, \gamma)) = \{w\}^*$. It is easy to see that this specification is equivalent to the no-deadlock specification we had in the context of Proposition 4.4. Therefore Problem C is also PSPACE-hard for the same reasons. ∎

References

[B] D. P. Bertsekas, *Dynamic Programming and Stochastic Control*, Academic Press, New York, 1976.

[CDFV] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya, Supervisory control of discrete-event processes with partial observations, *IEEE Trans. Automat. Control*, **33** (1988), 249–260.

[PS] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Opimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, NJ, 1982.

[PT] C. H. Papadimitriou and J. N. Tsitsiklis, The complexity of Markov decision processes, *Math. Oper. Res.*, **12** (1987), 441–450.

[P] C. P. Pfleeger, State reduction in incompletely specified finite-state machines, *IEEE Trans. Comput.*, **22** (1973), 1099–1102.

[RW1] P. J. Ramadge and W. M. Wonham, Modular supervisory control of discrete event systems, *Proceedings of the Seventh International Conference on the Analysis and Optimization of Systems*, Antibes, June 25–27, 1986, pp. 202–214, Lecture Notes in Control and Optimization of Systems, Vol. 83 (A. Bensoussan and J. L. Lions, eds.), Springer-Verlag, New York, 1986.

[RW2] P. J. Ramadge and W. M. Wonham, Supervisory control of a class of discrete-event processes, *SIAM J. Control Optim.*, **25** (1987), 206–230.

[T] J. N. Tsitsiklis, On the Control of Discrete-event Dynamical Systems, Technical Report LIDS-P-1661, Laboratory for Information and Decision Systems, M.I.T., Cambridge, MA March 1987.

[VW] A. F. Vaz and W. M. Wonham, On supervisor reduction in discrete-event systems, *Internat. J. Control*, **44** (1986), 475–491.

[WR] W. M. Wonham and P. J. Ramadge, On the supremal controllable sublanguage of a given language, *SIAM J. Control Optim.*, **25** (1987), 637–659.